5

INTERCONNECTION ARCHITECTURE AND METHOD OF ASSESSING INTERCONNECTION ARCHITECTURE

TECHNICAL FIELD

10 The present invention relates generally to the field of chip design and fabrication. The present invention also relates generally to the field of circuit routing.

BACKGROUND ART

15 With submicron technology, large numbers of processors, elements, or devices can be integrated on microcircuit chips. The processors, elements, or devices are arranged in arrays of cells on one or more layers of a chip. Each of the cells, containing a component of one or more overall circuits, contains one or more terminals for communicating with other cells. To permit
20 the cells to communicate with one another interconnects, such as routing wires or other conductive paths, connect the cells and/or bus segments, which themselves interconnect groups of cells.

The interconnects are arranged in meshes formed in or among one or more interconnect layers (also known as routing layers) of a microcircuit
25 chip. A mesh is a common routing architecture for many reconfigurable computing systems. Both conventional and more recently proposed on-chip multiprocessor systems use mesh networks as communication backbones.

The microcircuit chips typically include a plurality of interconnect layers for interconnection of the cells. Pluralities of layers are
30 often used for individual interconnections due to design constraints, for example. Vias help to route the interconnects between pluralities of layers.

Connections are switched by devices such as, but not limited to, metal oxide semiconductor (MOS) devices.

High-performance system-on-a-chip (SoC) requires nonblocking interconnects between the array of cells on the chip. With nonblocking
5   interconnects, when a cell needs to communicate with another cell, a route always exists for communication.

Interconnects have become one of the most precious resources on a chip. Length of connection between cells is a limiting performance factor in terms of power consumption and latency, among other factors. Unreasonable
10   distribution of interconnect resources results in bottlenecks that stall data flow, while leaving other routing resources wasted. Furthermore, it is impractical to resolve this problem merely by enlarging a channel capacity of an entire array.

A long path through interconnects increases power consumption and signal delay. Additionally, a common physical embodiment of
15   multiprocessor arrays is CMOS technology. In CMOS technology, power dissipation is proportional to interconnect capacitance, which in turn is proportional to a distance traveled by a signal. Thus, it is highly desirable to provide an architecture in which interconnection length is minimized. It is also desirable to provide an architecture that includes the shortest totals of route
20   lengths between processors, and not interconnect length alone.

One predominant type of interconnect mesh is Manhattan architecture, so-called because its rectilinear connection arrangement resembles a city street grid. Manhattan architecture, however, requires lengths of interconnects that far exceed actual (Euclidean) distances between individual
25   cells due to, for example, the requirement for orthogonal circuit paths.

More recently, an alternative chip architecture known as X-architecture has been introduced to reduce interconnection lengths versus Manhattan architecture. X-architecture uses tree structures having recursive patterns to interconnect cells in a nonblocking interconnection architecture.
30   The tree structures may take the form of H-shaped patterns or X-shaped patterns, with the cells located at the extremities of each pattern. The

2

interconnects are oriented, for example, in 0°, 45°, 90°, and 135° directions. X-architecture has been disclosed as a solution to address microcircuit chip designs, especially chips with five or more routing layers.

5   Interconnection between all cells is provided by a specific hierarchical structure. For example, at a level "zero", four cells may be interconnected by an "X". At a higher level, say, level "1", four level "zero" "X's" are interconnected by a larger "X". At a still-higher level ("2"), four level "1" "X's" are interconnected by a still-larger "X", etc. Performance improvement of the X-architecture over the Manhattan architecture has been

10   demonstrated.

DISCLOSURE OF INVENTION

The present invention provides, among other features, a multi-celled chip. The chip includes arrays of hexagonal cells arranged on at least

15   one component layer. A plurality of interconnects including Y's that connect the cells in clusters of three cells each. Each of the Y's has a node and three interconnects connecting the node to respective ones of the cells within a cluster, wherein each Y connects each cell of its respective cell group to the node.

20   The present invention also provides a number of methods to assess particular interconnection architectures, including providing a cost function and an assessment method based on a multi-commodity flow model.- Exemplary embodiments of chips and interconnection architectures are also provided that are selected using the assessment methods provided. Bridges are

25   also provided for directly connecting cells of a chip, and methods are provided for determining optimum locations of the bridges.

BRIEF DESCRIPTION OF DRAWINGS

FIG. 1 shows a chip having cells connected by Y-architecture

30   according to a preferred embodiment of the present invention, including a chart showing hierarchical levels of connection;

FIGs. 2A and 2B show cells connected by inverted and upright orientations of Ys, according to exemplary embodiments;

FIGs. 3A and 3B show two examples of six level Y-trees and their respective configurations;

FIGs. 4A and 4B show coordinates for cells in a hexagonal array and orientations for Ys, respectively, according to a preferred formation method;

FIGs. 5A and 5B collectively show an exemplary algorithm for expanding Y-trees according to a preferred method;

FIGs. 6A-6D show Y-trees formed as a result of the algorithm of FIGs. 5A and 5B and a tree representation for FIG. 6C;

FIG. 7 shows a polygon on the backbone of hexagons, with exemplary coordinates for polygon boundaries;

FIG. 8 shows two polygons having different orientations;

FIG. 9 shows an exemplary algorithm for merging two polygons according to a preferred method;

FIG. 10 shows a pair of merged polygons according to the exemplary algorithm of FIG. 9;

FIG. 11 shows an example of oriented merging of polygons according to another aspect of the present invention;

FIG. 12 shows an exemplary inventive method for merging three polygons of subtrees;

FIGs. 13A and 13B are plan and perspective views of a conventional via arrangement;

FIGs. 14A and 14B are plan and perspective views of a tunnel arrangement, according to an embodiment of the present invention;

FIG. 15 is a plan view of a bank of tunnels according to a preferred embodiment;

FIG. 16 shows a tunnel used to detour interconnects having a Y-architecture around a via according to an embodiment of the present invention;

FIG. 17 shows an exemplary bank of tunnels for a Y-architecture;

FIG. 18 is a schematic of a plurality of banks of tunnels and interconnects on different routing layers using Y-architecture according to another embodiment of the present invention;

FIGs. 19A-19C show prior art X-architecture model, a switch, and a larger X-tree, respectively, with an illustration of hierarchical levels;

FIG. 20 is a table showing Lx and Dx calculations for the X-architecture of FIGs. 19A and 19B, according to an exemplary cost function of the present invention;

FIG. 21 is a table showing Lx and Dx calculations for a Y-architecture model;

FIG. 22 is a schematic of cell structures in one dimension connected using fewer switches;

FIG. 23 are schematics of cell structures in one dimension connected using a greater number of switches, respectively;

FIG. 24 shows a cost function showing calculation of L, D, and M for the cell structures of FIGs. 22 and 23 according to a method of the present invention;

FIGs. 25A-25F show exemplary models of basic interconnect architectures;

FIG. 26 is a table showing calculation of L, D, and M for the models of FIGs. 25A-25F;

FIGs. 27A-27B show clusters of hexagonal cells connected using a Y and a triangular connection scheme, respectively, according to an embodiment of the present invention;

FIG 28 is a table showing calculation of L, D, and M for the models of FIGs. 27A-27F;

FIGs. 29A-29B show interconnect construction and a level diagram using the model of FIG. 25F;

FIGs. 30A and 30B show exemplary switches of the present invention having four and two inputs, respectively;

FIG. 31 shows a detoured routing between adjacent cells according to an embodiment of the present invention;

FIG. 32 shows the architecture of FIG. 30A with potential additional bridges according to an embodiment of the present invention;

FIG. 33 shows alternative additional bridges at five levels for the architecture of FIG. 30A;

FIG. 34 shows optimal bridges for the architecture determined according to a method of the present invention;

FIG. 35 is a table showing derivative benefits for bridges of different levels of the architecture of FIG. 30A according to a method of the present invention;

FIGs. 36A-36B show a construction of a prior art X-tree using the model of FIG. 25E;

FIG. 37 shows possible bridges for the model of FIG. 36A according to a method of the present invention;

FIGs. 38A-38B show a Y-architecture construction from hexagonal cells, having empty cells according to an embodiment of the present invention;

FIG. 39 shows a Y-architecture construction without empty cells, according to a preferred embodiment of the present invention;

FIG. 40 shows construction from a group of hexagonal cells, without empty cells, according to a preferred embodiment of the present invention;

FIG. 41 shows calculations of L and D for possible bridges for the model of FIG. 39;

FIGs. 42A-42B show locations of possible bridges for the model of FIG. 39;

FIGs. 43A-43C show possible bridges for Y-architectures without empty cells on levels 2, 1, and 0, respectively;

FIG. 44 is a schematic plan view of a conventional five-by-five communication mesh;

FIG. 45 is a graph representation of the communication mesh of FIG. 44 according to an embodiment of the present invention;

FIG. 46 is a schematic of a 45° mesh according to an embodiment of the present invention;

5   FIG. 47 is a schematic of a communication graph for the mesh of FIG. 25 according to an embodiment of the present invention;

FIG. 48 is a graph representation of the communication mesh of FIG. 2, with 45° interconnects added;

FIG. 49 is a schematic plan view of a communication mesh

10   having a hexagonal architecture according to an embodiment of the present invention;

FIG. 50 is a graph representation of the communication mesh of FIG. 49 being connected by routing wires having a Y-architecture according to an embodiment of the present invention;

15   FIG. 51 shows pseudo-code of a preferred multi-commodity flow (MCF) algorithm according to an embodiment of the present invention;

FIG. 52 shows a network flow model for multilayer routing according to an embodiment of the present invention;

FIG. 53 is a graph representation of a seven-by-seven

20   communication mesh being connected by interconnects having a Y-architecture, according to an embodiment of the present invention;

FIG. 54 shows a conventional seven-by-seven interconnect mesh using Manhattan architecture;

FIG. 55 is a graph representation of the communication mesh of

25   FIG. 54 being connected by interconnects having Manhattan architecture;

FIG. 56 shows a conventional seven-by-seven interconnect mesh using X-architecture;

FIG. 57 is a table showing throughputs of uniform edge capacity meshes according to an embodiment of the present invention according to an

30   embodiment of the present invention;

FIGs. 58A and 58B are graphs of n = 4 and 5 meshes, respectively, showing bottlenecks of communication flow;

FIG. 59 is a table showing throughputs of meshes having fixed edge capacities according to an embodiment of the present invention;

5   FIG. 60 is a table showing optimal capabilities for vertical edges in a 6 x 6 mesh according to an embodiment of the present invention;

FIG. 61 is a graph illustrating optimal sums of rows in a 9 x 9 mesh;

FIG. 62 is a table illustrating results of a 45° mesh according to

10   an embodiment of the present invention;

FIGs. 63A-63B are interconnect graphs showing flow congestion for 45° mesh structures, for n=5 and n=6 , respectively according to an embodiment of the present invention;

FIGs. 64A-64B are schematics showing vertical and horizontal

15   routing layers for 90° routing according to an embodiment of the present invention;

FIGs. 65A-65B are schematics showing routing layers for 45° routing according to an embodiment of the present invention;

FIG. 66 is a table showing throughputs for 45° and 90° mixed

20   mesh according to an embodiment of the present invention;

FIG. 67 is a schematic showing multiple routing layer assignments for a mixed 45° and 90° set of routing layers according to an embodiment of the present invention;

FIG. 68 is a table showing throughputs of the multiple routing

25   layer assignments of FIG. 38 according to an embodiment of the present invention;

FIG. 69 is an illustration of routing layers between two nodes, showing both Manhattan and diagonal routing directions according to an embodiment of the present invention;

FIG. 70 is a table showing throughputs of an n x n mesh using Manhattan architecture, Y-architecture, and X-architecture, respectively, according to an embodiment of the present invention;

FIG. 71 shows a congestion pattern of a twelve-by-twelve mesh using Y-architecture according to an embodiment of the present invention;

FIG. 72 shows a congestion pattern of a twelve-by-twelve mesh using X-architecture according to an embodiment of the present invention;

FIGs. 73A-73F show level two symmetrical meshes and communication graphs for Y-architecture, X-architecture, and Manhattan architecture, respectively, according to an embodiment of the present invention;

FIG. 74 is a table showing throughput of symmetrical hexagonal meshes connected using Y-architecture;

FIG. 75 is a table showing throughput of symmetrical octagonal meshes connected using X-architecture according to an embodiment of the present invention;

FIG. 76 is a table showing throughput of symmetrical octagonal meshes connected using Manhattan architecture according to an embodiment of the present invention; and

FIGs. 77A-77C illustrate the flow congestion patterns of a level 6 hexagonal mesh, a level 3 octagonal mesh, and a level 8 diamond mesh, respectively, according to an embodiment of the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

Interconnections among the cell array reveal themselves as a key problem, as the interconnect becomes one of the most precious resources on a chip. With the advent of deep sub-micron technologies, switches are becoming less costly, yet interconnects such as wires are still expensive. Therefore, optimization efforts according to embodiments of the present invention focus on the interconnect resources.

Traditional Manhattan interconnect architecture organizes interconnects on two orthogonal routing directions, 0° and 90°, for the

simplicity of routing embedding and design rule checking. However, this artificial restriction on routing directions adds significant interconnect length compared with the Euclidean optimum, and thus decreases the communication capability of the on-chip interconnects.

5   One goal of certain embodiments of the present invention is to allocate channel capacities in a mesh routing architecture to improve, or maximize, its communication capability. Communication capability can be measured by the throughput, which is the amount of information that every pair of nodes can exchange simultaneously. Throughput is a function of channel

10   capacity and the dimension of the processor array.

Chips have been disclosed including non-rectilinear interconnects to improve the efficiency of on-chip interconnects. Most of these chips have introduced 45° short jogs to improve routability of the chip in the detailed routing stage. Even in this architecture, however, the majority of the

15   interconnects on the chip have still been routed in directions of either 0° or 90 °.

As an alternative to the traditional, Manhattan architecture, Mutrunoi et al. proposed an on-chip architecture known as X-architecture, which is designed to target designs having five or more routing layers. I.

20   Mutsunori, T. Mitsuhashi, A. Le, S. Kazi, Y. Lin, A. Fujimiura, and S. Teig, "A Diagonal Interconnect Architecture and Its Application to RISC Core Design," Proc. ISSCC, pp. 684-689, San Jose, CA, Feb. 2002. In X-architecture, interconnects are arranged in 0°, 45°, 90°, and 135° directions. This design has been shown to achieve significant chip performance improvement and power

25   reduction over Manhattan architecture.

However, with X-architecture, it is possible for two nodes to be physically adjacent on a chip layer and yet be on different tree structures on the same level. Furthermore, these respective tree structures may be linked to separate tree structures on a higher level, or even a still-higher level, until a

30   level is reached, called a root, that is a common ancestor to the cells. Consequently, a greatly extended path length through interconnects may have

to be traversed to interconnect two cells even through they may be physically adjacent. It is desirable to gain still further improvement in performance, including power consumption and speed.

5   Another constraint on throughput of an active device array is the problem of getting a signal or power from one area, for example a quadrant, of a chip to another. To do so, a middle row or middle column in the interconnect mesh typically must be traversed. Due to the normal distribution of interconnections, a middle row or middle column of the interconnect mesh tends to create a bottleneck effect. Enlarging the congested area will not itself

10   produce better throughput. It is therefore desirable to provide an improved geometry to increase throughput.

According to an embodiment of the present invention, a configuration is provided in which an interconnect architecture includes one or more Y's to connect clusters of cells. A Y is a structural routing model in

15   which interconnects, or legs, extend in three separate directions from a common node. An architecture formed of Y's is termed herein a Y-tree, and allows interconnection among some or all cells in a hexagonal pattern. Groups of Y's routed together form Y-trees. In an exemplary embodiment, individual Y's on a particular level connect clusters of cells, and these Y's are

20   interconnected by Y's on higher levels. In the higher levels, a Y on a next-higher level is preferably rotated with respect to the Y on the next-lower level.

For example, an interconnect mesh having Y-architecture is provided in a multi-element integrated circuit chip array. Interconnects are routed in three directions, e.g. 0°, 60°, and 120°; or 0°, 120°, and 240°. The

25   mesh preferably comprises a plurality of layers. In an additionally preferred aspect, the cells are arranged in a hexagonal array and embodied in a chip having a shape of a convex polygon, such as a hexagonal chip. Individual Y's connect clusters of the hexagonal cells. Diagonal routing technology allows different arrangements of interconnect structure. Methods for fabricating

30   diagonal routing are provided in, for example, I. Mutsunori, T. Mitsuhashi, A. Le, S. Kazi, Y. Lin, A. Fujimiura, and S. Teig, "A Diagonal Interconnect

Architecture and Its Application to RISC Core Design," Proc. ISSCC, pp. 684-689, San Jose, CA, Feb. 2002.

In a preferred embodiment, the hexagonal cell array produces a flow congestion pattern that does not include the center of the hexagonal pattern. However, the benefit of producing a flow congestion pattern that does not include the center of the hexagonal pattern is not a function of any particular values of angles between the legs of individual Y's. Particular angles of the legs are not required; for example, 0°, 60°, and 120°, are merely an exemplary choice for artwork design. However, legs in one cell should be configured to connect with legs in a next cell. Wide tolerances between the specific values of the tree angles are allowed, while providing the same utility of the Y's. For example, a Y having legs at 0°, 150°, and 210° (forming a more traditional "Y" shape) could be provided.

The hexagonal cell array also has the property of hierarchical expansion. An algorithm is provided to set up a hierarchical tree of interconnect, and another algorithm is provided to set up a communication route in the architecture for pairs of processors in the array. It has been determined that the Y-architecture approaches the X-architecture in terms of optimizing wire resources. Additionally, algorithms for the merge of polygons on a hexagonal backbone are provided, which is useful in analysis of very large Y-trees.

According to an additional embodiment of the present invention, a cost function is provided to balance the cost of interconnect resources and the power consumption for the interconnect topology on a cell array. The total interconnect length is used to measure the cost of the interconnect resources, and the length of signal paths is used to evaluate the power consumption, since the power consumption is proportional to the interconnect capacitance, which in turn is proportional to the traveling distance of a signal.

An exemplary application of the cost function is used herein to compare shapes of meshes of cells. Each form of connection can be arranged in differently shaped meshes. For example, Manhattan architecture is most

readily arranged in a square mesh; however, it may also be embodied in a diamond-shaped mesh, which may be visualized as a square rotated by 45° from the position in which it rests on a side. Furthermore, the X-architecture lends itself to arrangement in an octagonal mesh, among other mesh shapes.

5   To provide geometry less susceptible to bottlenecks, embodiments of the present invention provide alternative polygonal meshes, which may be formed using dies, for example.

According to an exemplary application of the cost function, the X-type nonblocking architecture ("X-architecture") has been found to have a

10   good tradeoff for a two-dimensional processor array. A significant benefit to X-architecture is that it can be hierarchically expanded. This benefit has been shown to be applicable to Y-architecture as well. The X-architecture and Y-architecture, along with other architectures, can be compared using the provided cost function.

15   Methods also are provided for determining locations of optimal additional interconnects between certain cells, buses, and/or switches. These methods help to overcome some of the deficiencies in prior architectures, while continuing to require a minimum cost of interconnects and communication resources.

20   A method for assessing routing architecture is also provided. The Y-architecture of the present invention, having three routing directions, is compared with the Manhattan architecture and the X-architecture (with two and four routing directions, respectively). Using Y-architecture potentially gains a throughput improvement of 33.3% over the traditional Manhattan architecture

25   on a square mesh. The Y-architecture produces nearly the same (2.6%) throughput as the X-architecture on a square mesh, yet using one less routing direction.

Furthermore, the Y-architecture achieves an average of 13.4% interconnect length reduction over Manhattan architecture, and approaches

30   (4.3% less) the reduction of the X-architecture, while providing a simpler design. Still further, making the shape of the chip a convex polygon, and

preferably closer to a circle, significantly improves the throughput over the rectangular chip. Using Y-architecture, a hexagonal chip can produce 41% more throughput than a square chip using Manhattan architecture, without causing dead space on the wafer.

5   The described Y-architecture and other optimization methods are applicable not only to chip design, but to other areas such as, but not limited to, wireless communications. In an exemplary wireless communication design base stations may be seated at the centers of the hexagonal areas in an array, and a route between the base stations may form the main part of the wireless

10   communication route. The high performance solutions to communication among the base stations are quite similar to those of an array of processors on a chip. Therefore, the methods described herein are applicable to optimization of interconnect of base stations to balance cable resources and power consumptions.

15   Referring now to the drawings, FIG. 1 shows a chip 100 including an array 102 of hexagonal cells 104 (which may include processors or other chip components) interconnected through Y-architecture. The cells 104 have the physical shape of hexagons. Similar to X-architecture, the hexagonal cell array 102 can be expanded hierarchically.

20   As shown in FIG. 1, the array 104 is divided into clusters 106 of three cells 104. Every three cells 104 within the cluster on a level zero are interconnected with a Y 108 on a first level 110, as described above. Each Y 108 has three legs 112 of interconnects oriented in, for example, 0°, 120°, and 240° (symmetrical) routing directions, respectively, from a preferably central

25   node 114.

As also shown in FIG. 1, clusters of three nodes 114 of individual Y's 108 on the first level 110 are in turn clustered with a second level 116 of Y's. One level of the Y 108 is made up of at lease three routing layers, one for each direction. The number of routing layers for a level of the Y 108 can vary,

30   depending on how many layers are needed for each direction of the Y. Each of the second level 116 of Y's 108 in the embodiment shown is rotated 90° from

the Y's of the first level 110. By recursively arranging the Y's 108 in this manner, a hierarchical Y-tree 120 can be provided without empty or dead cells (cells not connected to a remainder of the Y-tree), as shown.

In the Y-tree 120 shown in FIG. 1, three subtrees (lower level Y-

5   shaped trees) are clustered on each level, and the length of the line segment, or

leg of a n-th level Y, $T_n = \sqrt{3}T_{n-1}$. At the second level, 3^2 hexagonal cells are bounded by the dashed line shown.

The Y-architecture preferably is routed upon the hexagonal array 102 including a number of rows. This array 102 preferably has the following

10   properties: (1) a ½ grid shift exists between rows; (2) each cell 104 is physically adjacent to two cells in the same row; and (3) each cell is physically adjacent to two cells in the neighboring row above and two cells in the row below. Depending on the orientation, these rules can be respectively applied instead to columns. Thus, groups of three neighboring cells 102 can be

15   clustered to set up individual Y's 108. Exemplary clusters 106 of three cells 104 are shown in bold in the hexagonal array of FIGs. 2A and 2B, in respectively inverted directions.

If each cluster 106 of three cells 104 is regarded as a unit, it can be seen that the hexagonal array 102 composed of such units also has the

20   property of ½ grid shift, but now in the vertical direction (in the orientation shown). Thus, the Y-architecture can be expanded to the second level 116; however, the directions of the individual Y's 108 at the second level 116 have a rotation of 90° compared to the Y's of the first level 110. In a preferred embodiment, this property of ½ grid shift always holds when the Y-architecture

25   is continuously expanded to upper levels. As shown in FIG. 1, for example, respective Y's 108 on higher levels are rotated either by 90° or -90° with respect to a previous (or higher) level.

For a Y-architecture of n tree levels, there are 2n combinations of orientations of the n Y's 108 on different tree levels. A combination of Y's

30   108 is referred to herein as a configuration, which indicates the way the overall Y-architecture grows. The configuration results in a particular boundary for

the cells 104 interconnected by the Y-architecture. FIGs. 3A and 3B show two examples 122, 124 of 6-level Y-architectures, together with their configurations.

Given a particular configuration, C, a Y-architecture as shown in FIG. 1 can be formed using the following expanding algorithm. The architecture is an ordered one. The first, second, and third subtrees correspond to three orientations as illustrated in FIG. 4B. Every node in the architecture, except the leaves (the nodes at individual cells 104), stores the orientation of the Y 108 with which to organize its three subtrees. C[1] denotes the orientation of the Y 108 on the lowest level, and C[n] denotes the orientation on the highest level. C[m] ∈ {up, down, left, right}. For consistency, but without limiting the scope of the present method, an exemplary configuration is started with an inverted Y. In other words, C[1] is assumed to equal "down". The coordinates shown in FIG. 4A distinguish the cells 104 in the hexagonal array. The center node 114 of the Y 108 at the highest level is shown in FIG. 4A, having coordinates (0, 0). FIG. 5 shows an exemplary algorithm implementing a design of a Y-tree based on these coordinates.

According to the exemplary algorithm Setup_Y_tree shown in FIGs. 5A-5B, the first three steps make subtrees of the first level Y tree from the leaves (i.e., the hexagonal cells 104). The fourth step makes the template Y tree for the first level from three subtrees according to the orientation C(1). Next, the fifth step calculates the coordinate shift base x, y, and z. The sixth step is a loop for all the levels in the Y-tree 120 from 2 to n. Within the loop, for level i, the coordinate shifts for the three subtrees, x1, y1, x2, y2, x3, y3, are calculated according to C(i), which is the orientation of the Y 108 at level i. Then, the template tree at the previous level is copied to be the three subtrees of the current level Y tree by a Copy subroutine. The coordinates of all the leaf nodes in the three subtrees are shifted by a Shift subroutine. Finally, the template tree for the i-th level is built based on the three subtrees and the orientation C(i).

FIGs. 6A-6D show an example for a Y-tree 120 formed by implementing the algorithm of FIG. 5. The configuration is "down, left, up". FIGs. 6A, 6B, and 6C are results for I=1, 2, 3, respectively. FIG. 6D shows a tree representation for the Y-tree shown in FIG. 6C, in which the coordinates in the leaves, from left to right, are:

(5,2)(4,1)(6,1)(2,3)(1,2)(3,2)(2,1)(1,0)(3,0)(-1,2)

(-2,1)(0,1)(-4,3)(-5,2)(-3,2)(-4,1)(-5,0)(-3,0)(2,-1)

(1,-2)(3,-2)(-1,0)(-2,-1)(0,-1)(-1,-2)(-2,-3)(0,-3)

From the hexagonal array's 102 properties and the algorithm for setting up the Y-trees 120, it can be shown that: (1) the exemplary algorithm according to an embodiment of the present invention generates Y-tree architecture without cell overlapping; (2) the number of cells covered by the generated Y-tree of n levels is 3^n; and (3) the length of the trunk at level n is $(1/\sqrt{3})^n$.

In another embodiment of the present invention, a merging algorithm is used to merge two polygons. Then, based on this algorithm, another algorithm for merging polygons to set up a Y-tree without empty cells is provided.

Suppose there exists a polygon 122 on a backbone of hexagons, as shown in FIG. 7. The polygon 122 can be represented with a sequence of integers, where every integer $i \in 0, 1$. In an exemplary embodiment, the sequence of integers is determined by traversing the boundary of the polygon in a counterclockwise direction. The boundary of the polygon 122 includes a series of adjacent edges 124. Every edge 124 has a rotation of either 120° or -120° with respect to its preceding edge. If an edge A has a rotation of 120° relative to its preceding edge, the edge 124 is considered to have a positive rotation. If the edge 124 has a rotation of -120°, the edge is considered to have a negative rotation. For example, in the polygon 122 shown in FIG. 7, edges a and b have positive rotations, while edge c has a negative rotation.

A sequence, termed herein a hexagonal sequence, can thus be determined to represent the polygon 122. Starting with the edge 124 of the

boundary of the polygon and traveling counterclockwise, if the edge 124 has a positive rotation, a 1 is entered into the sequence. If the edge 124 has a negative rotation, a 0 is entered into the sequence. The resulting string represents the hexagonal sequence. If A denotes a hexagonal sequence, then

5      $A(i)$ is defined to refer to the ith element in A, where $A(i) \in 0,1$.

For example, the hexagonal sequence of the polygon 122 shown in FIG. 7 is 110111011101, and the hexagonal sequence of each of the polygons a and b shown in FIGs. 8A and 8B is 11011011101101. Although the two polygons 122 have different orientations, the two polygons are considered

10   the same, as direction is not imposed.

It can be seen that one can make any bits barrel-shift (assumed herein, for consistency only, to be leftwards) on a non-oriented hexagonal sequence without changing the corresponding polygon 122. Furthermore, for a correct hexagonal sequence, the number of 1's will be six more than the

15   number of 0's, while for any sub-sequence, the difference between the number of 1's and the number of 0's should not exceed five. It can also be seen that two polygons 122 have the same shape and area (assuming unit size of cells) if they have the same hex-sequence. Additionally, if polygon 122 is flipped, its hex-sequence is also horizontally flipped. Thus, for a symmetric polygon, the

20   hex-sequence should be unchanged if the polygon 122 is flipped.

In an exemplary merging method, if rotations are not permitted for generation of Y-trees 120, a definition is assumed for an oriented hex-sequence. Every edge 124 on the polygon 122 thus has only three possible directions, and an oriented hexagonal sequence is denoted by starting the

25   hexagonal sequence with a vertical edge that is traversed downwardly. Therefore, the oriented hex-sequence for the polygon shown in FIG. 8A is 10110111011011, and the oriented hex-sequence for the polygon in FIG. 8B is 10111011011101.

The direction of each edge 124 can be calculated easily according

30   to the numbers of "1's" and "0's" ahead of the edge. For an oriented hexagonal sequence A, i bits can be made to barrel-shift on the oriented hex-sequence

without changing the direction of the polygon 122 if, and only if, the difference between the number of "1's", and the number of "0"s is either zero or five in the subsequence from A(2) to A(i+1).

Given two hex-sequences A and B, an algorithm may be used to
5   provide a new hex-sequence C, which is a merging of polygons A and B. To merge the hex-sequences, it is first assumed that both polygons A and B can be rotated. A preferred embodiment of the algorithm is shown in FIG. 9. In the algorithm, the first step retrieves the bit-wise complement, BI, of the input sequence B. The second step generates the reversed bit order sequence, Bh, of
10   sequence BI. Then, for each common sub-sequence, sub, between Bh and A, if it is acceptable for merging by the Accept function, the following is performed: Rewrite sequence A in the form A=(A1)(sub)(A2); rewrite sequence B in the form B=(B1)RevFlip(sub)(B2), where RevFlip(sub) is an operation on sequence sub to complement every bits followed by a bit order reversing;
15   calculate sequence A12 = ModMerge(A1, A2), and B12 = ModMerge(B1, B2), where ModMerge(S1, S2) is an operation to merge two sequences S2 and S1 and get sequence S = (S2)(S1), and then to complement the first bit of S and eliminate the last bit of S. The sequence of the merged polygon, C, is the sequence A12 followed by sequence B12.

20   FIG. 10 gives an example of merged polygons 122 illustrating use of the algorithm. This example finds the merging with the longest adjacent boundary. For hex-sequence A = 110111011101 and B = 0011101110101111,- BI (Bit-wise complement B) = 1100010001010000, and Bh (bit order reversed BI) = 0000101000100011, Then:

25   sub = 110; RevFlip(sub) = 100
             A1 = 1101;
             A2 = 11101
             A12 = 01101110
             B1 = 1110111010111
30   B2 = empty
             B12 = 011011101011

C = (A12)(B12) = 011011100110111101011

Next, the situation is considered in which the polygons 122 are not allowed to rotate. In other words, two oriented polygons 122 are merged. The design of the function "Accept" in the algorithm of FIG. 9 should be slightly more complicated. Not only should the two subsequences have the same pattern, but they also should have the same directions for their corresponding edges 124. In addition, if the common sub-sequence happens to involve the first bit of A, or A1 has the only bit 1, the first bit of B is the first bit of the merged polygon C, and the generated hexagonal sequence must be shifted to make it correctively denote the polygon's orientation. It can be shown that the first bits of A and B will not appear in the sub-sequence simultaneously. A polygon formed by oriented merging is shown in FIG. 11.

In implementing this algorithm, A = 1100111011101011; and B = 101110111011 (remembering the position of the first bit in B). Thus:

BI = 010001000100

Bh = 001000100010

sub = 100; RevFlip(sub)=110

A1 = 1

A2 = 111011101011

A12 = 011011101011

B1 = $\hat{1}$011101

B2 = 11

B12 = 01$\hat{1}$01110

C = 0110111010110$\hat{1}$01110 (C needs an orientation adjust)

C = $\hat{1}$01110011011110101101

The final polygon 122 of a complete Y-tree 120 can be obtained by merging the polygons of sub-trees, from the lowest level to the highest. When two polygons 122 are merged, they have a section of common boundary. The two ends of the common boundary may be connected with a line, in which the direction of the line is defined to be the direction of the common boundary.

Given the direction of the first edge 124 of the common boundary and the corresponding sub-sequence, the direction of the common boundary can be easily calculated. Merging of three polygons 122 of sub-trees can be realized, under the orientation configuration of Y at each level, by two steps:

5   (1) merge two of the three polygons with the following two conditions satisfied: (a) the length of the common boundary is one-sixth the length of the original polygon's boundary; and (b) the direction of the common boundary should be vertical or horizontal, depending on the required orientation of the Y;

10   (2) merge the polygon generated in step (1) and the remaining original polygon, with the following two conditions satisfied: (a) the length of the common boundary is one-third the length of the original polygon's boundary; and (b) if the common boundary is split into two halves, the directions of the two halves should be opposite to the required orientation of Y.

15   For simplicity, one starts from the sub-tree of level 2. A subtree of level 2 includes 9 basic hexagonal cells 104 and has a completely symmetrical polygon regardless of the directions of the Y's on the first and second levels. FIGs. 12A and 12B are illustrations of merged polygons formed by steps (1) and (2) above, respectively. After step (1), a common boundary

20   126 length is = 4, which is one-sixth the boundary of each original polygon. The common boundary's direction 126 is vertical. After step (2), the common boundary's 126 total length is 8 (again, one-third the original polygon size), and as split into two halves, each half of the common boundary has a direction opposite to that of the required orientation of Y for level 2 as shown. By

25   merging the polygons of sub-trees level by level, we get the final polygon 122 of the Y-tree 120. Preferably, the process of merging will not result in empty cells.

In multilayer routing, a via is used to connect interconnects that are disposed on multiple layers. However, the via blocks wire tracks on layers

30   it passes through. According to another embodiment of the present invention, tunnel detours are used to route interconnects around vias.

21

FIGs. 13A and 13B show a horizontal interconnect 130 in a horizontal (x) direction routing layer. A path of the horizontal interconnect 130 is impeded by a gap having a via 132, which extends in a top-to-bottom (z) direction through a number of routing layers. An exemplary structure is shown

5   in FIGs. 14A-14B to address this problem. As shown, the horizontal interconnect is connected to a tunnel 134, which extends in the z-direction down to a lower routing layer, in this case, a y-direction routing layer, so that the tunnel includes another x-direction interconnect spanning the gap, but displaced from it in the y-direction due to a pair of y-direction interconnect.

10   The generally U-shaped tunnel 134, as can be more clearly seen in FIG. 14B, allows the horizontal interconnect to avoid the vias 132 within the gap.

To maximize throughput, a plurality of tunnels 134 preferably forms a bank 138, which is arranged on a lower routing layer along a direction of a plurality of gaps and vias 132. As shown in FIG. 15, suppose L is the

15   dimension of the bank 138 and c is the number of vias 132 in each individual tunnel 134, the number of vias avoided is equal to cL. The top n-k layers in this embodiment are used to distribute signals to the bank 138. In this configuration, on the top n-k layers, $c+2$ wiring tracks are blocked on each vertical layer while all the wiring tracks on the horizontal layers can be routed

20   without blockage.

FIG. 16 shows an example of a tunnel 134 for Y-architecture. The via 132 blocks tracks on a layer of 60° direction for a plurality of interconnects arranged according to a Y-architecture. As shown, for interconnects 130 in the 90° and 120° directions, via tunnels 134 are provided

25   to detour the interconnects around the via. For example, a first entry 120° interconnect is routed through the tunnel 134 in the 60° layer, passing underneath a third 120° wire, and then exits. Similarly, a first 90° interconnect avoids the via 132 on its own, 90° routing layer, but a second 90° interconnect avoids the detoured first 90° interconnect through another tunnel in the 60°

30   layer.

The pattern formed by the five interconnects shown in FIG. 16 with their respective tunnels 134 can be extended to form the bank 138 of tunnels, as in the case of Manhattan architecture. An exemplary bank 138 of tunnels is shown in FIG. 17. A plurality of banks 138 of tunnels is shown in FIG. 18. Again, the bottom k layers are used to perform intra-cell routing, and the top n-k layers are used to distribute signals to the banks 138 of tunnels. If L is the dimension of the bank and c_l is the number of vias 132 in each individual via tunnel 134, the number of vias in this arrangement preferably equals $c_l L$, and the number of overhead tracks preferably equals $c_1 + c_2$, where c_2 is the constant associated with the individual tunnel design. In the exemplary tunnel 134 shown in FIG. 17, c_1 and c_2 are equal to 1 and 5, respectively. The banks of via tunnels maximize throughput.

The design of early blocking networks focused on minimization of switches. In deep submicron technology, devices are shrunk to very small sizes and are less expensive, while interconnects such as wires and buses are lengthened, resulting in the increase of interconnect resistance and capacitance. Performance such as power consumption and signal delay are significantly deteriorated. Therefore, the length of signal paths is more important than the number of switches in the path regarding delay in circuit processing. However, a large-scale system on a chip (SoC) requires a significant amount of wire resources, so it is not feasible to set up the shortest path for every pair of processors in the array.

Conventionally, bus-based architectures have offered standards for communication interfaces. However, in chip design, a length of connection between cells is a limiting performance factor in terms of power consumption and latency, among other factors. The physical size of long interconnects limits the scalability of the architecture. Also, the contention for the interconnects adds to the latency of the communication. This increase is made more significant by the ever-shrinking size of individual cells and interconnects (in width, for example). Thus, chip designs minimizing connection lengths provide a performance benefit for a particular chip.

The total length may be used to measure the cost of interconnect resources, because the interconnect length typically is proportional to the amount of area taken on the routing layers. In deep submicron technologies, the number of routing layers remains limited. Furthermore, even as the number of routing layers increases, the coupling capacitance due to congestion and the required vias that connect signals to the layers high above make routing area a precious resource. It is also desirable to reduce the power dissipation of the wire interconnects because power consumption has become one of the main concerns in various applications.

According to a preferred method of the present invention, an objective cost function is provided to balance interconnect topology between routing area and power dissipation. This cost function is defined with an emphasis on interconnects as opposed to switches.

A goal of the cost function is to reduce the total traveled distance of the signal communication. Let us assume that each cell has to communicate with the rest of the cells with equal demand. Then, the total power dissipation is measured by the total pairwise distance between the cells. This equal demand model is used for preferred embodiments of the present method because the demand is symmetrical and thus independent of the placement implementation.

It is conceivable that by adding interconnects for the communication, the traveling distance can be reduced. However, the interconnects resources are limited by the physical space. Furthermore, the same resources are needed for other purposes such as, but not limited to, making internal connections within each cell, or for testing. Thus, the product of the total interconnect length and the total power distribution is chosen as a metric to balance design. Moreover, the derivative of this product provides an additional metric to further analyze the interconnect architecture.

A preferred method for determining benefit of a particular tree structure thus includes minimization of a cost function, as shown below.

Min $M = L * D$

Where, $L = \sum$ Length of each wire

$$D = \sum_{i \leq i < P} d_{i,j}$$

In this definition, $d_{i,j}$ is the shortest route length between processor (i) and processor (j). P is the total number of processors.

5   In conventional hierarchical interconnection architectures, either L or D of the cost function above has been minimized at the expense of increasing the value of the other parameter. The Y-structure of the present invention, on the other hand, helps to minimize or substantially reduce M. In a preferred integrated circuit design method of the present invention, the cost

10   function is utilized for various configurations, and a configuration that minimizes M may thus be selected for design of an integrated circuit.

For rectangular cells, X-architecture provides optimal performance according to the above cost function. FIG. 19A shows an X-architecture model in which an X 142 having four legs of interconnects connect

15   each of a 2 x 2 array of cells 140. The center of the X 142 includes a switch box 144, which has the internal structure shown in FIG. 19B. As shown in FIG. 19B, the switch box 144 includes six switches 146 that may connect the four intersected interconnects. Every two of the four cells 140 covered by the X-architecture can set up a communicating route through the switch box 144.

20   The interconnects from the four cells 140 are also bundled together, forming a new interconnect going to a higher level, as shown in FIG. 19C. A higher level X-tree 148 also includes a larger switch box 144 (indicated by a larger circle) lying at higher levels, which has a similar structure of the switch box of FIG. 19B, except that the bus width grows four

25   times for every expansion to a higher level. This architecture guarantees that whenever two processors in the array need to communicate, a route always exists.

Assuming the distance between the cells 140 is equal to one, the table of FIG. 20 shows application of the above-described cost function to the

30   X-architecture of FIGs. 19A-19C. N denotes the highest level of the X-

architecture as shown in the X-tree 148 of FIG. 19C. It is preferred that the tree 148 grows recursively, and that the topology of each level remains identical, with certain rotations allowed. The total number of cells 140 covered by the X-tree 148 preferably is equal to k^n, where k is the number of the

5   subtrees clustered on each level of the X-architecture. The length of the trunk at level n, T_n, preferably is equal to $\sqrt{k}T_{n-1}$. It is also preferred that no "holes" or "empty cells" exist within the array. In other words, it is preferred that the cells 140 in the array form a continuous region bounded by a closed curve.

Assuming that the distance between the centers of adjacent cells

10   140 is equal to α, the key results of the cost function as applied to the Y-architecture are shown in the table of FIG. 21.

In an exemplary method comparing Y-architecture to X-architecture, it is assumed that the cells 140 in the two architectures have one unit area. Thus, the distance between the centers of adjacent cells 140 in X-

15   architecture is one, and the distance between the centers of adjacent cells in the Y-architecture, α in the table of FIG. 21, is $3^{-1/4} \cdot 2^{1/2}$. FIG. 22 demonstrates functions of M_X and M_Y with respect to n, the highest level of the architecture. In FIG. 22, M_X and M_Y are normalized with A^4, where A is the number of cells 140 that the tree covers.

20   To make a comparison for greater n levels, we neglect the lower order items of M_X and M_Y:

$$M_X \approx \left(6 \cdot 2^{4n-4} 2^n \sqrt{2}\right)\left(2^{3n-2}\sqrt{2}\right) = 6 \cdot 2^{8n-5}$$

$$M_Y \approx 3^{n-1}\sqrt{3^n}\left(3+\sqrt{3}\right)\left(3^{\frac{n}{2}}-1\right)3^{2n-2}\alpha^2 = 2\left(1+\sqrt{3}\right)3^{4n-3}$$

To compare the respective performance of the trees

25   mathematically, we assume that the trees 120, 148 cover the same number of cells 140. This results in:

$$n_X = \log_4 A \qquad n_Y = \log_3 A$$

$$M_X = \frac{6}{2^5}A^4 \qquad M_Y = \frac{2(1+\sqrt{3})}{3^3}A^4$$

$$\frac{M_X}{M_Y} \approx 0.93$$

A is the number of cells 140, 104 in the X- or Y-tree. As shown, there is just a slight difference between M_X and M_Y. Therefore, the two architectures have similar performance. If A is close to some order of four, X-

5   architecture is a preferred solution, while if A is close to some order of three, Y-architecture is preferred.

The derivative form of the cost function may also be used to further analyze the interconnect architecture, and is given by:

$$\frac{\Delta M}{\Delta L} = \frac{(L + \Delta L)(D + \Delta D) - L * D}{\Delta L}$$

10

$$= \frac{\Delta D}{\Delta L} + \Delta D + D$$

$$\approx \frac{\Delta D}{\Delta L} L + D$$

The last equation is based on the assumption that $\frac{L}{\Delta L}$ is much larger than one. To identify the most cost-effective incremental improvement due to the change of L, a derivative benefit is provided. The derivative benefit

15   I is:

$$I = -\frac{\Delta D}{\Delta L}$$

A negative sign is used because D is expected to decrease when L increases.

Based on examples of the cost function, one-dimensional, two-

20   dimensional, and three-dimensional nonblocking interconnect architectures can be compared, and preferred structures can be selected. An embodiment of the present invention provides, among other things, a hierarchical interconnection architecture in which bridges are provided between physically proximate nodes that may otherwise be distant via interconnect routing. The bridge is preferably

25   placed between nodes on the same level. A method is provided to select an optimum level on which to provide a bridge. Making a bridge between nodes

perturbs the tree structure, and an optimal solution is derived in terms of derivative benefit.

FIGs. 22 and 23 show an array of cells 140 arranged along a straight line and connected by interconnects 130 in the form of buses. In FIG. 22 switches 146 are shown (by an "x") between pairs of crossing horizontal and vertical interconnects 130. When the switch 146 is on, the vertical interconnect connects to the horizontal interconnect. Otherwise, the two interconnects are not connected. In FIG. 22, groups 144 of six switches 146 (shown as non-filled circle) represents connect six possible pairs of the four ports. Thus, the architecture of FIG. 23 requires more switches, but less wire resources, than the architecture of FIG. 22.

Applying the above cost function to the architectures of FIGs. 22 and 23, the results for L, D, and M are shown in FIG. 24, assuming that the distance between adjacent cells 140 is 1. Thus, using the cost function, the architecture of FIG. 23 is preferred for one-dimensional non-blocking architecture, because it has the minimum number of interconnects necessary to connect the two parts separated by the cutline, and because, for every pair of cells, the shortest signal route is provided.

Similarly, the cost function can be applied to two-dimensional architectures. FIGs. 25A-25F show a number of nonblocking interconnect architecture models including rectilinear interconnects (also referred to as "Manhattan interconnects") and/or diagonal interconnects connecting a 2 x 2 array of cells 140, and their associated switches 146. The models shown in FIGs. 25A-25C have a mesh structure, the models shown in FIGs. 25E-25F have a tree structure, and the model shown in FIG. 25D has a mixture of mesh and tree structures. FIG. 26 shows cost function values from each of the models shown in FIGs. 25A-25F.

Though the model shown in FIG. 25B is a subset of the interconnect set of the architecture in FIG. 25A, FIG. 26 shows that the total pairwise distance D is the same. The set of interconnects of the model of FIG. 25C is a subset of that of the model of FIG. 25B. However, the model of FIG.

25C has a total pairwise distance D longer than that of the model of FIG. 25B. The quality of the two models is equal in terms of the cost function.

Models of FIG. 25D and FIG. 25E both adopt 45° interconnects. The model of FIG. 25D has less total pairwise distance D than that of the 5 model of FIG. 25E. However, the model of FIG. 25D requires a much longer length of interconnects. Thus, according to the cost function, the model of FIG. 25D is worse than that of FIG. 25E.

The model of FIG. 25F uses an H-tree topology. Since the interconnects are forced to follow a Manhattan pattern, the quality of the model 10 of FIG. 25F is worse than that of FIG. 25E. Finally, the model of FIG. 25D has the minimum pairwise distance D, for it provides the shortest signal route for any pair of cells 140, but the model of FIG. 25E consumes the fewest overall interconnect resources, and is the preferred model of those shown in FIGs. 25A-25F according to the cost function.

15   FIGs. 27A and 27B both show an architecture model for a physical layout of hexagonal cells 104. The model of FIG. 27A shows an interconnection of a Y-type, in which each of the group of three cells 104 is connected to a central switch node by diagonal interconnects. The model of FIG. 27B, by contrast, has a triangular architecture, in which each of the cells 20 104 are not connected to a central point, but rather at the nodes at each cell by a triangle 160. Assuming that the distance between the centers of neighboring cells 104 is 1, by employing the cost function, results of which are shown in- FIG. 28, it is apparent that the model for using the Y 108 is preferred to the model of the alternative triangular architecture based on the objective function, 25 as it has a significantly lower overall wire length M.

According to certain embodiments of the present method, the cost function described above can be applied to improve particular interconnection architectures. For example, FIGs. 29A – 29B illustrate an H-tree 162 based on the H-architecture model shown in FIG. 25F. As shown in FIG. 29A, the cells 30 140 are connected by interconnects 130 and switches 146, and in addition, two interconnects of the same level are bundled together to form a new interconnect

connection to the level above. Switch groups 144 are shown in FIG. 30 for four and two inputs, respectively. The interconnection width doubles for every expansion to a higher level. The expansion continues until the root of the tree is reached. FIG. 29B describes the hierarchy of the tree structure and the

5   definition of levels 164 for the tree 162. To form a square array, the number denoting the tree's 162 top level, n, must be even. The number of cells 140 covered by the tree with 162 n levels is equal to $P_n=2^n$.

A principal shortcoming of this structure is the extra detouring problem. An extreme example of this is depicted in FIG. 31. Two cells 140

10   may be close in geometric distance, but their actual interconnection route can be much longer if their lowest common ancestor of a hierarchical tree structure is the root.

To reduce this shortcoming, and according to an embodiment of the present invention, interconnections referred to herein as bridges 170 are

15   added to connect (bridge) nodes of the same level. As shown in FIG. 29A, the terminals 166 of each of the cells 140 and the switches 146 at the interconnects are potential nodes for interconnection. The communication between the bridged nodes can thus bypass the detour of going toward upper levels by taking advantage of the bridge 170. FIG. 32 shows exemplary locations of

20   bridges 170 between pairs of nodes.

A preferred method of choosing optimum locations of the bridges 170 is provided. Given an n-level tree structure, for each integer m ($0 \leq m \leq$ n), the incremental improvement of level-m nodes is stated as follows.

(1) Two level-m nodes (the T joints of the H tree) are considered

25   physically adjacent if the Euclidean distance between the pair is the closest among all level-m nodes.

(2) A pair of level-m nodes is connected if the nodes in the pair are physically adjacent and if their lowest common ancestor of the tree structure is the root. Level-m nodes are linked with 2^m buses.

30   FIG. 33 illustrates the alternative bridges at five levels using an array of 8 x 8 cells 140 as an example. Only the upper half array is shown. In

FIG. 33, the tree structure depicted in FIG. 29A is eliminated to clarify the illustration. The additional wires are symmetrical with respect to the central vertical line that divides the cell array into halves.

A question is then presented as to the level for establishing the bridges 170 to obtain the largest benefit. To resolve this, a derivative benefit function is derived according to the derivative benefit defined above. Given a tree of level n, and the level investigated m:

$$\Delta D(n,m) = A(n,m) * B(n,m)$$

In this equation, A (n, m) represents the number of pairs of cells 140 that will benefit from the addition of the bridges 170, and B (n, m) represents the route length saved due to the bridges. Thus, if m is odd:

$$A(n,m) = 2^{\frac{n+3m-1}{2}}$$

$$B(n,m) = -\left(2^{\frac{n+2}{2}} - 2^{\frac{m+3}{2}}\right)$$

$$\Delta L(n,m) = 2^{\frac{n+2m-2}{2}}$$

$$I(n,m) = 2^{\frac{n+m+3}{2}} - 2^{m+2}$$

For example, in the architecture of FIG. 29, if m=n-1, I=0 because B (n, n-1) = 0. If m is even:

$$A(n,m) = 2^{\frac{n+3m}{2}}$$

$$B(n,m) = -\left(2^{\frac{n+2}{2}} - 3*2^{\frac{m}{2}}\right)$$

$$\Delta L(n,m) = 2^{\frac{n+2m}{2}}$$

$$I(n,m) = 2^{\frac{n+m+2}{2}} - 3*2^{m}$$

For any even m (0 < m < n):

$$I(n,m) - I(n,m) \qquad = 2^{\frac{n+m+2}{2}} - 3*2^{m} - 2^{\frac{n+m+2}{2}} + 2^{m+1}$$

$$= -2^{m} < 0$$

From the above inequality, it can be shown that I (n, m) < I (n, m-1). Hence, in this example, only the odd levels are inspected for maximum derivative benefit. For a continuous variable function: $I(n,x) = 2^{\frac{n+x+2}{2}} - 3*2^x$, we calculate that when x = n-1, the derivative benefit is maximized, $I(n, n-3) = 2^{n-1}$.

Thus, for an H-tree architecture, level m = n - 3 gives an optimal derivative benefit for bridges 170: $I_{max} = 2^{n-1}$. FIG. 34 shows a number of optimally placed bridges 170, shown in dashed lines. FIG. 35 shows the derivative benefits for different levels (values of I (n, m)). As shown, the best solution is neither at the highest level nor at the lowest level.

In another example, the bridges 170 are added to the X-tree architecture model according to FIG. 25E. The hierarchical extension is shown in FIG. 36A. The bus width expands four times for every migration to a higher level. The dashed line 172 represent a connection to the similar arrays in the chip 100. For the case in which the tree's root lies at the n-th level, the number of processors is $P_n = 4^n$.

The bridges 170 are added to the architecture of FIG. 36. FIG. 37 is a top portion of an 8 x 8 cell array according to the architecture of FIG. 36, illustrating exemplary alternatives for bridges 170 at different levels. Again, the X-tree architecture of FIG. 36 has been removed from FIG. 55 for clarity. The additionally connected nodes 114 are preferably all symmetrical with respect to the large cross that divides the entire cell array into four parts.

Given an n-level X-tree structure, and using the method described above, incremental improvements are considered by using the bridges 170 to link nodes 114 at different levels. For each level m: $0 \le m < n$, pairs of level-m nodes 114 are connected if the pairs are physically adjacent and their lowest common ancestor in the X-tree is the root. Level-m nodes 114 are linked with 4^m interconnects 130. The derivative benefit is derived as follows:

$$A(n,m) = 4*2^{n-m-1}*4^m*4^m = 2^{n+3m+1}$$

$$B(n,m) = -\left(\sqrt{2} * \sum_{i=m}^{n-1} 2^i - 2^m\right)$$

$$\Delta L(n,m) = 4 * 2^{n-m-1} * 2^m * 4^m = 2^{n+2m+1}$$

$$I(n,m) = \sqrt{2} * 2^{n+m} - 2^{2m} * \left(\sqrt{2} + 1\right)$$

For   the   continuous   variable   function:

5    $I(n,x) = \sqrt{2} * 2^{n+x} - 2^{2x} * \left(\sqrt{2} + 1\right)$, there exists an $x_0 = 1 < n - x_0 < 2$, such that I (n,

x_0) has a maximum value. Further calculation shows that I (n, n-2) > I (n, n-1).

Therefore, it is preferred that, for an X-tree architecture, level m = n - 2 gives

the best derivative benefit for additional interconnects: $2^{2(n-2)}[2^{5/2} - (\sqrt{2} + 1)]$.

Another example of providing the bridges 170 is given with

10   respect to Y-architecture. FIG. 38 shows one type of Y-tree architecture. In

the Y-tree architecture of FIG. 38, each of the Y-trees 120 is oriented in the

same direction. As shown, there is a plurality of dead cells 178 (shaded in FIG.

56), indicating that some cells are excluded from the wire interconnect covered

by the Y-tree.

15   FIGs. 39 and 40 show levels 0-3 and 4-5, respectively, of an

example of another type of Y-tree architecture, in which there are no dead cells,

but the orientations of Y's 108 are rotated with each increase of tree levels.

The table of FIG. 41 shows values of L and D for Y-trees 120 with n levels.

For a large n, M_{no_empt} (FIG. 39) has a smaller value than M_{with_empt} (FIG. 38),

20   and is thus preferred.

However, the rotation of Y's 108 presents additional difficulty for

adding bridges 170. The interconnection architecture that is shown in FIG.

42A indicates examples of bridges 170 on the Y-architecture of FIG. 38.

Regardless of the level considered, the number of possible bridges 170 was the

25   same, all equaling three The possible bridges 170 connect adjacent nodes 114

that otherwise are connected only at the root.

The optimization method described above can be used to

determine the derivative benefit for a Y-architecture with dead cells 178.

$$A(n,m) = 3^m * 3^m * 3 = 3^{2m+1}$$

$$B(n,m) = -\left(\frac{2}{\sqrt{3}} * \sum_{i=m}^{n-1} 2^i - 2^m \right)$$

$$\Delta L(n,m) = 3^{m+1} * 2^m$$

$$I(n,m) = \sqrt{3} * 3^{m-1} * \left(2^{n-m-1} - 2 \right) - 3^m$$

The optimal level on which to put the bridges 170 is level n-2,

5   with the maximum incremental benefit: $I(n,n-2) = (2\sqrt{3}-1)*3^{n-2}$.

If, instead, the bridges 170 are placed on level n-1, the top level Y

108 can be removed, as shown in FIG. 42B. Then, ΔL becomes

$3^n * 2^{n-1} - \sqrt{3} * 6^{n-1}$, resulting in the incremental benefit of $1/2*3^{n-1}(\sqrt{3}-1)$.

However, this is still less than I (n, n-2).

10   FIGs. 43A – 43C show exemplary bridges 170 on a Y-

architecture without dead cells, at levels 2, 1, and 0, respectively. Using the

optimizing function:

$$A(n,m) = 3^m * 3^m * 3 = 3^{2m+1}$$

$$B(n,m) = -\left(\frac{2}{\sqrt{3}} * \sum_{i=m}^{n-1} \sqrt{3^i} - \sqrt{3^m} \right)$$

15     $$\Delta L(n,m) = 3^{m+1} * \sqrt{3^m}$$

$$I(n,m) = 3^m \left(\frac{2}{3-\sqrt{3}} \left(\sqrt{3^{n-m}} - 1 \right) - 1 \right)$$

Again, the maximum incremental benefit is provided at level n-2.

Accordingly, for Y-tree architecture with or without the dead cells 178, it is

preferred that level m = n-2 is used for a location of the bridges to provide the

20   best derivative benefit for the bridges. $I_{max} = 3^{n-2}(2\sqrt{3}-1)$ for the architecture

with dead cells 178, and $I_{max} = 3^{n-2}(2\sqrt{3}+3)$ for the architecture without dead

cells.

Thus, in an exemplary implementation of the cost function and

derivative benefit, it can be determined that, when adding the bridges 170 to X,

25   H, and Y tree structures, the incremental improvement connecting nodes at 2,

3, and 2 levels, respectively, below the root are optimal.

It is advantageous for chip design optimization to focus on the interconnect resources. In the future, significantly greater numbers of routing layers (for example, twelve or more) will be available in high performance circuit designs. Thus, it is desirable to consider various ways to organize on-chip routing resources. However, the prohibitive cost of actually designing and manufacturing a chip with new interconnect architectures makes it difficult to implement and test new interconnect architectures individually. Thus, it is highly desirable to develop a quantitative framework to evaluate the efficiency of different interconnect architectures.

In prior methods of evaluating efficiency, the interconnect length reduction was studied by allowing more routing directions, but all of these methods concentrated on the Steiner cost of a single signal net. Competition over routing resources between different nets is typically ignored using these methods.

According to another aspect of the present invention, an assessment method for determining a benefit of a particular structure is provided. This method adopts a multi-commodity flow (MCF) approach to model the on-chip communication traffic. MCF is a natural way to model communication network traffic. For example, MCF has been used to study wide area communication network traffic. However, due to the high computing complexity of MCF, most uses of this approach adopt heuristic methods to approximate an MCF solution.

A preferred embodiment of the present assessment method extends the MCF algorithm to solve various MCF problems and provides improved chip routing design methods. Solution of MCF finds the optimal throughput for a given routing architecture.

According to a preferred method of the present invention, stated generally, a mesh structure is assumed having uniform communication demand; that is, the routing demand is equal for every pair of nodes. The MCF throughput of the mesh structure is used to measure communication capability of different interconnect architectures. This method is independent of

particular test cases, and is independent of placement and routing. The extended MCF according to a preferred assessing method can reflect the exact communication bottlenecks on the chip or network, and it can provide a feasible upper bound of communication.

Algorithms involving this type of MCF can be solved fairly efficiently using, for example, the methods described in N. Garg and J. Koneman, "Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems," In Proc. Of the 39th Annual Symposium on Foundations of Computer Science, pp. 300-309, 1998.

Turning now to an exemplary assessment method, FIG. 44 shows a five-by-five communication mesh 180 connected using Manhattan architecture. For Manhattan architecture, communication resources for a group of cells are decomposed into an array of $n \times n$ slots 182. Each slot 182 contains a communication terminal, for example, a processor. The mesh 180 of FIG. 44 is an example of a 90-degree mesh structure with twenty-five slots 182. The slots 182 are aligned in rows and columns. Each square tile represents a slot. The mesh structure 180 can be mapped to a graph $G = \{V, E\}$, as shown in FIG. 45, according to the following rules:

(1) Each slot 180 i corresponds to the node 186 i in the graph.

(2) The adjacency between two slots 182 (i, j) is represented by an edge 184 e = (i, j) in the graph.

(3) The edge capacity c (e) is proportional to the length of the line segment separating the adjacent slots 182, and the number of routing layers.

A uniform communication requirement is assumed; that is, every pair of nodes 186 communicates with an equal demand. All communications are assumed to happen at the same time. The model can be extended to various other communication demands as well such as, but not limited to, Poisson distribution, Rents rule, etc., depending on specific applications. For simplicity and for generalness, the example of uniform pairwise communication is adopted for the description herein. Uniform pairwise communication demand

also provides an unbiased symmetry, which makes the solution independent of the test cases, placement, and routing.

Throughput, z, is defined to be maximum amount of communication flow between every pair of nodes 186. The throughput is determined using a MCF model. The flow that starts from node i is defined as "commodity" i. Commodity i starts from node 186 i with the amount of z (N - 1), where $N = n^2$ is the number of nodes in the graph, to each of the rest of the nodes with the amount of z. The MCF problem is solved to find the maximum throughput z.

The above MCF problem can be formulated as a linear program in either the node-arc form (LP1), or the edge-path form (LP2). The node-arc form (LP1) of MCF is:

$$\textit{Maximize} : z$$

$$S.t. \quad \sum_{j \in neighbor\ of\ i} (f_{ji}^v - f_{ij}^v) = \begin{cases} -z \cdot (n^2 - 1) & \textit{if } i = v \\ z & \textit{otherwise} \end{cases} \quad \text{for all nodes } v, i \in V$$

$$\sum_{v \in V} (f_{ij}^v + f_{ji}^v) \le c_{ij} \quad \text{for all edges } (i,j) \in E$$

In this linear program, flow variable f_{ij}^v represents the flow amount of commodity v on edge 184 (i, j). The edge capacity c_{ij} represents the flow capacity of edge 184 (i, j), in a uniform mesh using X-architecture, and c_{ij} is set to be unitary for all (i, j). The flow injecting to a node 186 is set to be positive and the flow ejecting from a node is set to be negative.

This linear program includes two sets of constraints. The first constraint describes the flow conservation of each commodity v at each node 186 i. The second constraint denotes that the total amount of flow on each edge 184 is no more than the capacity of that edge.

The edge-path form of MCF (LP2) is as follows:

$$\text{Maximize: } z$$

$$\text{S. t.: } \sum_{p \in P_{ij}} f(p) - z \ge 0 \quad \text{for nodes } i, j \in V, i \neq j$$

$$\sum_{p \in Pe} f(p) \le c(e) \quad \text{for all edges } e \in E$$

In linear program LP2, P_e denotes the set of all paths p containing the edge 184 e, and P_{ij} denotes the set of all paths between nodes 186 i, j. The flow variable f(p) represents the flow amount of path p.

The number of linear constraints in linear program LP1 is $|V|^2 +$

5   $|E|$. Thus, the linear program LP1 can be solved in polynomial time using any polynomial time linear program solver, such as that disclosed in N. Karmarkar, "A new polynomial-time algorithm for linear programming," Combinatorica, 4(4):373-395, 1984. However, when n increases, the number of linear constraints significantly increases (at the rate of n^4 for an n x n mesh).

10   Thus, for large cases, it may be impractical to solve the MCF using linear programming.

A combinatorial $(1+\varepsilon)$-approximation approach has been proposed to solve the MCF problem. An example of this combinatorial approach is disclosed in N. Garg and J. Konemann, "Faster and Simpler Algorithms for

15   Multicommodity Flow and other Fractional Packing Problems," In Proc. of the 39th Annual Symposium on Foundations of Computer Science, pp, 300-309, 1998.

In an embodiment of the present invention, the approach of this approximation algorithm is extended to incorporate edge capacities as

20   variables. This approach adopts the primal-dual structure of the linear program LP2.

Generally stated, a preferred algorithm according to the present invention assigns a nonnegative shadow cost to each edge 184, according to the congestion level at that edge. Initially, all of the shadow costs are set to be

25   equal. Then, the algorithm proceeds in iterations. In each iteration, a fixed amount of flow is rerouted along the shortest path for every commodity. At the end of each iteration, the capacity of every edge 184, and its shadow cost, is adjusted according to the dual linear program.

For every given error tolerance ε, a preferred embodiment of this MCF algorithm can find a (1+ε) approximation of the throughput in

$$O\left(\frac{1}{\varepsilon'}\log_{1+\varepsilon'}\frac{n}{1-\varepsilon'}n^4\log n\right)$$ time, where $\varepsilon = 1 - (1+\varepsilon)^{-\frac{1}{3}}$.

In a preferred embodiment of the approximation method, all fractional flows are used. The throughput, $\hat{z}$, of the fractional flow model, is an upper bound of the throughput, $\tilde{z}$ of the integer flow model. However, networks such as a packet switching network in RAW and Smart Memories, do not require that the flow be an integer. For wire switching networks in FPGA's, the flow amounts can be interpreted as the number of wires, which need to be integers.

In R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995, pp. 79-83, it was shown that by randomized rounding, with the probability of 1-ε, one can find $\hat{z}$ approaches $\tilde{z}$ with inequality $\hat{z} \geq \tilde{z}/(1+\Delta^+(1/\hat{z}, \varepsilon/2N))$, where N is the number of nodes in the mesh, ε is any real number between 0 and 1, and Δ+(1/$\hat{z}$, ε/2N) is the value of δ such that

$$\left[e^{\delta}/(1+\delta)^{1+\delta}\right]^{1/\hat{z}} = \varepsilon/2N.$$

The MCF algorithm described above will now be used by example to compare throughput of a number of different mesh structures: the 90° mesh 180, a 45° mesh 190, and the 90° and 45° mixed mesh 192. Results show that the 45° mesh 190 can achieve better throughput than the 90° mesh 180. Moreover, 90° and 45° mixed mesh 192 can further improve throughput.

In a first set of examples of a preferred assessment method, a number of routing algorithms are constructed having different capacities and routing orientations. The first three structures are 90° meshes 180 with different edge capacities. In the first architecture, every edge 184 has a unitary capacity. In the second architecture, edges 184 on the same row or column have equal capacity. In the third architecture, edge capacities are flexible, but the sum of the capacities of all of the edges 184 is fixed. The fourth

architecture is a 45° mesh 190 where interconnections are routed at 45° angles. The fifth architecture is a mixture of 90° and 45° mesh 182. The sixth architecture is a mixed 90° and 45° mesh 192 with different routing direction assignments.

5   For the model of uniform edge capacity, all the edge capacity is set to a unit, that is, $c_{ij}=1$ for all edges 184 (i,j) in the graph. This case is used as a basis. It is assumed that the n x n array of slots 182 is evenly distributed in a square area.

           In the second interconnection structure, edge capacities c_{ij} are set

10   as variables. However, the capacities of edges 184 in the same row are set to be equal. Likewise, the vertical capacities of edges 184 in the same column are set to be equal. The sum of the vertical edge capacities in a row is set to be n, and the sum of the horizontal edge capacities in a column is set to be n. In other words, the height and width of the array remain n.

15   Let c_{Hi} be the capacity of horizontal edges 184 in the i-th row, and c_{Vi} be the capacity of vertical edges in the i-th column. We add the 2n variables, c_{H1}, c_{H2},..., c_{Hm}, c_{V1}, c_{V2}, c_{Vm}, to the linear program. The height and width constraints of the array can be expressed as:

$$\sum_{i=1}^{n} c_{Hi} = n \text{ and } \sum_{k=1}^{n} c_{vk} = n$$

20   For this structure, it is assumed that one can adjust the row height and the column width of the array of processors.

           For the third structure we give the program more freedom to choose the best edge capacities. We require only that the total capacity of all edges be a constant. This structure represents the best edge capacity we can

25   allocate for a 90° mesh. The resultant throughput is an upper bound of a 90° mesh architecture.

           We set the edge capacities, c_{ij}, as variables. The total capacity constraint is expressed as:

$$\sum_{for\,all\,edges\,i,j} c_{i,j} = 2 \bullet \left(n^2 - n \right)$$

40

Note that $2 \bullet \left(n^2 - n\right)$ is the number of edges 184 in an n x n mesh. For this structure, we assume that the area of each slot 182 is flexible. We adjust the height and width of each individual slot so that the total area remains the same.

5   The fourth structure adopts the 45° mesh 190. All interconnects are oriented in 45° or 135° directions. The size of the mesh 190 increases with n. For a 45° mesh 190 of n, the number of nodes 186 is $n^2 + (n - 1)^2$, and the number of edges 184 is $4 (n - 1)^2$. FIG. 46 shows an example of 45° mesh 190 of $n = 5$. FIG. 47 illustrates the graph corresponding to the mesh 190. In this

10   structure, we assume that the slots 182 are shaped in diamonds (a square rotated by 45°) and are aligned in 45° and 135° directions. Thus the edge capacity remains a unit; that is, $e_{ij}=1$.

In the fifth structure, we add diagonal edges; that is, 45° edges and 135° edges, to the 90° mesh 180 of Manhattan architecture to form the

15   structure represented by the communication graph shown in FIG. 44. FIG. 48 illustrates an example of the mixed mesh 192 for $n=5$. FIG. 44 shows the slot arrangement. Mixed 90° and 45° meshes 192 allow more freedom on routing directions. For an n x n mixed mesh, the number of nodes 186 is n^2 and the number of edges 184 is $2(n-1)^2 + 2(n^2-n)$.

20   As shown in FIG. 48, the edges 184 are oriented in 0°, 90°, 45°, or 135° angles. All nodes 186 are aligned in rows and columns. Thus, all rectilinear edges 184 in the 45° and 135° directions have the same capacity, and all of the diagonal edges in the 0° and 90° directions have the same edge capacity. The length of the diagonal edge 184 in the 45° direction or 135°

25   direction is $\sqrt{2}$ times that of the rectilinear edge in the 0° or 90° directions. Thus, if routing a number of interconnects on one of the rectilinear edges 184 consumes one unit of routing area, then routing the same number of interconnects on the diagonal edges would consume $\sqrt{2}$ units of routing area.

In other words, for a pair of routing layers, if a capacity of x can

30   be allocated to the rectilinear edges 184, only a capacity of $x/\sqrt{2}$ can be

41

allocated to the diagonal edges. If we let c_1 be the capacity of the rectilinear edges 184 and c_2 be the capacity of the diagonal edges, the area constraints can be expressed as $c_1 + \sqrt{2}c_2 = 1$. In this way, the total area is equal to the constant area of uniform structure.

FIG. 49 shows a hexagonal mesh 200 including a number of hexagonal cells 104 according to a chip embodiment incorporating an alternative, triangular embodiment of Y-architecture. FIG. 50 shows a corresponding communication graph. In FIG. 50, all of the edges 184 are symmetrically oriented in 0°, 60°, and 120° directions, and every edge has the same length. Accordingly, the routing area constraint for this embodiment of Y-architecture can be expressed as $c_1 + c_2 + c_3 = 2$, where c_1, c_2, and c_3 are the edge capacity for edges 184 oriented in 0°, 60°, and 120° directions, respectively.

The above routing area constraint can be added into the linear programs LP1 or LP2, treating the edge capacities as variables. The optimal solution of the linear program produces an optimal routing resource allocation for different routing directions. The routing resource allocation problem can be formally formulated in the following way:

Input: communication graph G = (V, E), k different routing channels $\{R_1, ..., R_k\}$, where $\bigcup_i R_i = E$ and $\bigcap_i R_i = \Phi$: edge capacity c_1 for every edge in the routing channel R_i and area constraints $\sum_i \alpha_i C_i = 1$

Output: a routing resource allocation $\{c_i\}$, such that the communication graph G = {V, E} has maximum throughput.

The routing resource allocation problem can be written as the following linear program:

$$Min: \sum_i \alpha_i C_i$$

$$S.t. \sum_{p=P_{i,j}} f(p) \geq 1 \qquad \text{for all distinct vertices pair } i, j \in V$$

$$\sum_{p \in P_e} f(p) \leq C_i \qquad \text{for all edge } e \in R_i$$

This linear program finds the minimum routing area that can satisfy the unit pairwise communication demand. The dual program of this linear program is:

$$Max: \sum_{ij} \lambda_{i,j}$$

5       $$S.t. \quad \lambda_{i,j} \le \sum_{e \in P_{i,j}} d_e \qquad \text{for all distinct vertices pair } i, j \in V$$

$$\sum_{e \in R_i} d_e \le \alpha_i \qquad \text{for all routing channel } R_i$$

The dual program assigns a nonnegative shadow cost d_e to each edge 184 e, such that the sum of the shortest distances between every distinct pair of nodes 186 is maximized. The constraints in the above equations denote

10   that the total shadow costs of all edges 184 in a routing channel are smaller than or equal to the area coefficient of that routing channel.

By extending the combinatorial (1+ε)-approximation scheme as described above, the routing resource allocation problem can be solved. In a preffered method, a shadow cost is determined by the flow congestion level on

15   each edge 184. Let $g(e)=(f(e))/(c_e)$ be the congestion level of edge 184 e, where f(e) is the total flow amount going through edge e, and c_e is the capacity of e. The shadow cost d(e) is computed using:

$$d(e) = \frac{\exp(\beta(g(e)-g))}{\sum_{e' \in E} \exp(\beta(g(e')-g^*))}, \text{ where } g^*=\max\{g(e)|e \in E\}, \text{ and } \beta \text{ is a}$$

constant related to desired approximation error ε.

20   Initially, all of the shadow costs are set to be equal. Then, the algorithm proceeds in iterations. In each iteration, a fixed amount of flow is rerouted along the shortest path for every commodity. At the end of each iteration, the capacity of every edge 184 and its shadow cost is adjusted according to the dual linear program. FIG. 51 shows exemplary pseudo-code

25   of the routing resource allocation algorithm.

The assessment algorithm will now be used to compare the Manhattan architecture, the Y-architecture, and the X-architecture for both rectangular and symmetrical chip designs. Vias 132 become an important

concern when the number of routing layers increases. An embodiment of the present invention provides a network flow model that considers the vias 132. The basic assumption made is that each via 132 will block one routing track. For each slot 182, we set an upper bound on the total number of vias 132 and

5   interconnects across the node 186.

For example, suppose there are k routing layers. Each slot 182 is now represented by k routing cells as shown in FIG. 52. Each routing cell includes two nodes 186: n_a and n_b. Node n_a takes all of the incoming edges from the neighboring routing cells, and node n_b ejects edges to neighboring

10   routing cells. An edge 184 with capacity c directs from node n_a to node n_b. This edge 184 is used to restrict the total number of vias 132 and interconnects crossing the routing cell. Using this flow model, we compare the communication throughputs with different routing layer assignments using the MCF model.

15   To assess performance of the above-described MCF method, we used Matlab's linear program package on a Sun Ultra10 workstation to compute MCF solutions. For a case with 100 nodes, the run time exceeds 24 hours. We then implemented the MCF algorithm and the above-described routing resource allocation algorithm using C programming language. The

20   implementation derived the MCF solutions for cases with up to 289 nodes within 12 hours.

Using the present routing resource algorithm, we compared the throughput of n x n meshes 210 using Manhattan architecture, Y-architecture, and X-architecture. FIG. 53 shows a seven-by-seven mesh using hexagonal

25   cells 104, and FIG. 54 shows an interconnection graph of the mesh of FIG. 53 using Y-architecture. FIGs. 55A and 55B show a seven-by-seven mesh 210 and interconnection using a rectilinear mesh and Manhattan architecture, and FIGs. 56A and 56B show a seven-by-seven mesh and interconnection using a rectilinear mesh and X-architecture. For an n x n mesh, the enclosing box of

30   the slots 182 is close to a rectangle. The throughput of an n x n mesh using a

particular interconnect architecture demonstrates the communication ability of that interconnect architecture on a rectangular chip.

For an n x n mesh with Y-architecture, there are $3n^2-4n+1$ edges; for an n x n mesh 210 with Manhattan architecture, there are $2n^2-2n$ 0° and 90°

5   edges; and for an n x n mesh with X-architecture, there are $2n^2-2n$ edges on 0° or 90° edges and $2(n-1)^2$ edges in the 45° or 135° direction. To fairly compare the throughput of meshes with different interconnect architectures, the same amount of routing resources should be allocated to meshes having the same size.

10   FIG. 57 shows the results of uniform edge capacity meshes with n = 2 to 20. The table shows the number of nodes 186 and throughput z. From this result, at least the following conclusions can be drawn:

-The throughput is $1/n$ when n is odd and $(n^2-1)/n^3$ when n is even.

15   -The throughput is limited by edges 184 on the middle column and row. When n is an even number, edges in the central row and column form the bottleneck of the flow. When n is an odd number, the two columns and two rows form the bottleneck. FIGs. 58A and 58B show the bottleneck of communication flow for n = 4 and 5, respectively. The congested edges 212

20   are marked with bold lines. Note that the bottlenecks form the horizontal and vertical cut sets. The cut lines 214 are shown in FIGs. 58A and 58B as dashed lines.

For example, for equal n, the throughput of a 90° mesh with uniform row and column capacities is exactly the same as that of the 90° mesh

25   with fixed edge capacities. No throughput improvement is obtained because the total capacity of the edges in each column and row is fixed.

For n = 2 to 10, FIG. 59 shows the results of 90° mesh with fixed total edge capacities. The fourth column provides the throughput improvement compared to that of 90° mesh with uniform edge capacity. As the total capacity

of each row or column is no longer limited, the average throughput improves 29.7% from n = 4 to 10.

The results also show that all edges 184 are congested. The optimal edge capacity is no longer uniform. The capacity is larger for the edges in the middle row and middle column. FIG. 60 shows the optimal edge capacities for all the vertical edges in a 6 x 6 mesh. The sum of all the capacities in each row is listed. FIG. 61 illustrates the optimal sums of the rows in a 9 x 9 mesh. Note that there are eight rows of vertical edges in a 9 x 9 mesh. Thus, the chip area is no longer a square, but a convex area.

FIG. 62 shows the results of a 45° mesh for n = 2 to 12. To compare the results in FIG. 62 and FIG. 57, we use the cases with almost the same number of nodes 186. For example, both the case of n=4 in FIG. 62 and the case of n=5 in FIG. 57 contain 25 nodes. The case with 45° mesh achieves the throughput of 0.209, which gains a 4.18 percent improvement. Also, we compare the case of n=7 in FIG. 62 with the case of n = 9 in FIG. 57. The case in FIG. 62 contains 85 nodes, which has 4 more nodes than the case in FIG. 57. The throughput of the 45° mesh case is 0.1260, which is 13.16% more than that of the 90° mesh case.

As shown in FIGs. 63A and 63B, the congested edges 212 also present a different pattern, in that they form four cut sets at four corners. FIGs. 63A and 63B show the flow congestion in 45° mesh for n=5 and n=6, respectively. The congested edges 212 are in bold lines, and the cut lines 214 are in dashed lines.

FIGs. 64A-64B and FIGs. 65A-65B illustrate why 45° routing is preferred to 90° routing. Assume that we have a square-shaped chip 220 with two routing layers. FIGs. 64A-64B illustrate the case of 90° routing and FIGs. 65A-65B depicts the case of 45° routing. A cut line 214 is shown for the horizontal congested edges in FIGs. 64A-64B. Only the interconnects on the horizontal routing layer could cross the cut line and the number of interconnects across the cut line is d/D, where d is the interconnect pitch and D

is the dimension of the chip 220. A similar cut line 214 is drawn in FIGs. 65A-65B. The number of edges 184 across the cut line in each layer is $d/\sqrt{2}D$. The total number of interconnects crossing the cut line for the two layers in FIGs. 65A-65B is $\sqrt{2}d/D$. Thus, the upper bound of throughput increases to

5    $\sqrt{2} = 1.414$. However, the throughput is now limited by the cut edges at four corners.

FIG. 66 depicts the results from the 90° and 45° mixed mesh structure. Column 2 lists the throughput z. Column 3 lists throughput improvements over the 90° meshes with uniform edge capacity. Columns 5

10   and 6 list the best capacity for horizontal and vertical edges, c_1, and the best capacity for 45° edges, c_2, respectively. Column 7 lists the normalized capacity ratio of the diagonal edges to the Manhattan edges.

At least the following observations can be made with regard to FIG. 66:

15   - The throughput of the mixed mesh 192 is better than the 90° mesh 180, given the equal communication resource. The improvement in the throughput is up to 20.04% for a large number of nodes. The improvement is also better than 45° mesh 190 in terms of throughput.

- With n increasing, the optimal ratio for the capacity of the 45°

20   edge to the 90° edge approaches 5.6.

Using the MCF model in FIG. 52, one can compute the optimal routing direction assignment for mixed 45° and 90° routing. Assume that there are four routing layers, and each of them is assigned to a different routing direction. FIG. 67 shows four different routing layer assignments. The

25   throughputs under four different assignments are listed in FIG. 68. As shown, the throughputs with assignments IV and I are about 16% larger than the throughputs with assignments II and III.

FIG. 69 illustrates why interleaving the Manhattan routing layers and diagonal routing layers can produce better throughput. As shown in FIG.

30   69, given two points (nodes 186) on the plane, the shortest way to connect them

is always a Manhattan line plus a diagonal line. Thus, if the Manhattan routing layer and the diagonal routing layer are interleaved, the interconnects can go along the shortest paths without a cost of more vias. This will produce better throughput.

5   In an exemplary comparison, the sum of all edge capacities is set to be equal to $2n^2-2n$ for all n x n meshes, and the routing resource algorithm is used to find the optimal allocation of edge capacities. FIG. 70 shows throughputs of n x n meshes for Manhattan architecture, Y-architecture, and X-architecture, respectively, for n from 2 to 17. The throughput was normalized

10   using a factor $m^{0.5}(m-1)$, where m is the number of nodes in the mesh. By doing so, the total amount of communication demand and total edge capacities are kept independent of the dimensions of the mesh. The third and fourth columns of FIG. 70 show throughput and normalized throughput of meshes using Manhattan architecture. The fifth and seventh columns depict the

15   normalized throughput of meshes using Y-architecture and X-architecture, respectively. The sixth and the eighth columns list the determined throughput improvement achieved by Y-architecture and X-architecture, respectively, over the Manhattan architecture.

        As shown in FIG. 70, for n from 10 to 17, Y-architecture

20   provides an average improvement of 30.7% for an n x n mesh, and X-architecture achieves a 34.5% improvement. For a 17 x 17 mesh, Y-architecture provides a throughput improvement of 31.1 % and X-architecture achieves an improvement of 34.6 %. Additionally, for Y-architecture and Manhattan architecture, equally distributed edge capacities produce maximum

25   throughput on n x n meshes. For X-architecture, the optimum ratio of the area of diagonal routing edges to that of Manhattan edges 184 is shown in the far right column of FIG. 70. This ratio approaches 5.65 when n increases.

        FIGs. 71 and 72 show bottlenecks of communication flows for 12 x 12 meshes using different interconnect architectures. The fully saturated

30   edges 282 are shown using bold lines. As shown, the saturated edges form vertical and horizontal cut sets for both interconnection architectures. The cut

lines 214 are shown as a symmetrical half using dashed lines. By summing the capacities of the edges passing across the cut lines 214, a throughput upper bound for n x n meshes with different interconnect architectures can be derived.

For example, for Manhattan architecture, there are n edges 184
5   crossing each cut line. The total edge capacity is n. For Y-architecture, there are 2n-1 edges 184 passing across each cut line 214, and each edge has capacity 2/3, so that the total edge capacity crossing the cut line is (4n-2)/3. When n approaches infinity, an n x n mesh using Y-architecture can have (4/3-1) – 33.3% more flow crossing the cut line 214. Thus, Y-architecture can achieve
10   up to 33.3% throughput improvement over Manhattan architecture on a squared mesh.

For X-architecture, there are 2(n-1) diagonal edges and n Manhattan edges crossing each of the two cut lines 214. To achieve maximum throughput, the ratio of the capacity for diagonal edges and the capacity for
15   Manhattan edges is 5.6. Under this ratio, the edge capacities are 0.1515 and 0.6 for the Manhattan edges and diagonal edges respectively. The total flow amount that can go across the cut line is $1.3535n$-1. When n approaches infinity, the throughput improvement bound is thus 35.6%.

For all of the cases that have been tested (n = 2 to 17), these kind
20   of central horizontal cut sets were observed using X-, Y-, and Manhattan architectures. Furthermore, in all of these cases, there is no flow passing through the same cut set more than once. If this is true for all n x n meshes, the improvement upper bounds derived are exact throughput improvement rates.

The same analysis was performed on symmetrical chip shapes as
25   described above. A rectangular chip has communication bottlenecks on its respective two middle cut lines. The physical dimension of the middle part of the chip restricts the communication flow, and thus prevents larger throughput. Using a convex-shaped chip, better throughput is possible by allowing more wires to cross the original middle cut lines. This is verified using an
30   embodiment of the routing algorithm of the present invention.

As shown in FIGs. 73A-73F, a shape of the chip 100 is designed to be a convex polygon as close as possible to a circle and symmetrical to all routing directions. The throughput of the different structures was then compared. FIGs. 73A and 73B show a level 2 hexagonal mesh 230, which is the symmetrical structure corresponding to the Y-architecture. FIGs. 73C and 73D illustrate an octagonal mesh 232, which is the corresponding symmetrical structure to the X-architecture. Finally, FIGs. 73E and 73F show a diamond-shaped mesh 234, which is symmetrical to the Manhattan architecture.

Using the above-described routing algorithm, throughput of the symmetrical structures 230, 232, 234 for the Y-architecture, X-architecture, and Manhattan architecture was computed. FIG. 74 shows the throughput of hexagonal meshes 230 from level 1 to level 7. FIG. 75 shows the throughput of octagonal meshes 232 from level 2 to level 4. FIG. 76 shows the throughputs of diamond meshes 234 from level 1 to level 12. Normalized throughputs by total edge capacities are also shown in FIGs. 74-76.

As shown, for Y-architecture, a hexagonal mesh 230 with 169 nodes, for example, produces 17.3% more throughput than a 13 x 13 rectangular mesh using the same interconnect architecture. For X-architecture, an octagonal mesh with 101 nodes, for example, can achieve 13.4% more throughput than a 10 x 10 rectangular mesh, which has 100 nodes. For Manhattan architecture, a diamond-shaped mesh 234 with 265 nodes, for example, provides a throughput of 5.61e-4, while a 16 x 16 mesh using the same interconnect architecture, which has 256 nodes, produces a throughput of 4.88e-4, so that a throughput of diamond mesh 234 over square mesh for Manhattan architecture is determined to be 15%.

As shown in FIGs. 77A-77C, the meshes with symmetrical structures produce different flow congestion patterns from n x n meshes. FIGs. 77A-77C illustrate the flow congestion patterns of a level 6 hexagonal mesh 230, a level 3 octagonal mesh 232, and a level 8 diamond mesh 234, respectively. The cut edges 212 are marked using bold lines. The symmetrical

meshes 230, 232, 234 display a more evenly distributed congestion pattern than n x n meshes. The middle cut lines do not exist any more.

The following exemplary benefits are thus revealed via the MCF algorithm of a preferred embodiment of the present invention:

5   -For uniform capacity mesh, the congested edges 212 lie in the center rows and columns. The total throughput of each node 186 is inversely proportional to the dimension of the mesh.

-The re-arrangement of capacities between different columns or rows will not improve the throughput if the total capacity of the columns or
10 rows is kept constant.

-A flexible chip shape provides a throughput improvement of about 30% over a square chip of equal area.

-A 45° mesh structure 190 produces about 17% more throughput over a 90° mesh 180 for a processor array of 144 nodes.

15 -A mixture of 90° and 45° mesh structures 192 can achieve an additional 30% throughput. To achieve maximum throughput, the ratio of resources allocated to the 45° routing layers versus those to the 90° routing layers approaches 5.6 as the number of nodes 186 increases.

-In the 90° and 45° mixed routing, interleaving the diagonal
20 routing layer and the Manhattan routing layers can reduce the number of vias and hence increase the communication throughput.

Interconnect length has a significant impact on virtually every important measure of chip quality. From the physical point of view, decreasing inteconnect length directly reduces the resistance and capacitance of the
25 interconnect, thus improving the performance and power consumption of the circuits. From a designer's point of view, shorter total interconnect length produces less routing congestion on the chip, and therefore improving the routability and signal integrity of the design. At the same time, from a manufacturing perspective, shortening the interconnect length can improve the
30 manufacturability and reliability of the chip.

Because of its highly limited freedom for choosing routing directions, Manhattan architecture adds a significant amount of interconnect length versus the Euclidean optimum. Allowing more routing directions has been found to shorten the total interconnect length. Previously, researchers

5   have studied the impact of using different interconnect architecture on the interconnect length. Many of these efforts have involved constructing the Steiner routing trees under different routing direction restriction. However, due to the inherent difficulty of the Steiner minimum tree problem, a significant amount of time has been spent developing heuristics for construction Steiner

10   trees for a randomly generated net, and for statistically calculating the average interconnect length for different interconnect architectures.

An additional embodiment of the present invention derives a quantitative comparison of interconnect lengths needed to connect a two pin net using different interconnect architectures. To generalize the non-rectilinear

15   routing structure, the concept of λ-geometry has been introduced. λ represents a number of possible routing directions. In λ-geometry, interconnects with angles $i\pi/\lambda$, for all i are allowed, where λ is a positive integer. λ = 2, 3, 4 correspond to the Manhattan architecture, Y-architecture, and X-architecture, respectively.

20   The derivation adheres to the following rules:

(1) In λ-geometry, given two points A and B, if AB are not on any of the λ feasible routing directions, then the shortest path connecting AB consists of two segments AC and CB, where the angle between AC and CB is $(1-1/\lambda)\pi$.

25   (2) Let A, B be any two points on the place, r_e be the Euclidean distance between A and B, and r_λ be the length of the shortest interconnect to connect AB in λ-geometry, then $\max_{A,B} \frac{r_e}{r_\lambda} = \csc\left(\left(\frac{\lambda-1}{2\lambda}\right)\pi\right)$.

(3) Let A, B be two random points on the plane, r_e be the expected Euclidean distance between A and B, and r_λ be the expected length of

52

the   shortest   interconnect   to   connect   AB   in   λ-geometry,   then

$$r_\lambda = \frac{2\lambda(1 - \cos(\pi/\lambda))}{\pi \sin(\pi/\lambda)} r_e.$$

Rule (1) provides that, in order to connect two pins with the shortest interconnect, there is at most one turn on the path, and it is desirable to maximize the angle between two segments of the path for the given interconnect architecture. For different interconnection architectures, Rule (2) determines the worst-case amount of additional interconnect length cost versus the Euclidean distance. For example, for Manhattan architecture, in the worst case, the interconnect length is 41.2 % longer that the Euclidean distance. For Y-architecture and X-architecture, respectively, the additional interconnect length is at most 15.47 % and 8.23 %.

Rule (3) determines the average interconnect length of a two pin net using different interconnection architectures. For Manhattan architecture, the average interconnect length is 27.32% longer than its Euclidean distance. For Y-architecture, the average interconnect length is 10.27% longer than its Euclidean distance. The X-architecture further reduces the average interconnect length to be within 5.48% of the Euclidean optimum and it produces 4.3% interconnect length reduction over Y-architecture, but with the added cost of one more routing direction.

A novel non-blocking hierarchical interconnect architecture, Y-architecture, has been shown and described herein. The hexagonal cell arrays employed in Y-architecture have the property of hierarchical expansion and therefore nonblocking hierarchical interconnect architectures can be set up on them. According to an objective function also provided herein to balance interconnects resources and performance, it is shown that Y-architecture preferably is only 7% less effective than X-architecture. Because the distribution of hexagonal cells has the same pattern as that of the base stations of wireless communication systems, the architecture provided herein can also be used to optimize wireless systems, for example.

While various embodiments of the present invention have been shown and described, it should be understood that other modifications, substitutions, and alternatives are apparent to one of ordinary skill in the art. Such modifications, substitutions, and alternatives can be made without

5   departing from the spirit and scope of the invention, which should be determined from the appended claims.

Various features of the invention are set forth in the appended claims.